Lab 5: Multiple regression

Statistics for Social Scientists II

Bur, GJM

2024-10-07

1 Multiple regression

The workflow for these notes is as follows: work on the first question in small groups. Then, I will reintroduce the subject space presentation of regression. This perspective is slightly tricky at first, but it makes it extremely easy to understand what the sums of squares are (squared lengths of centered variables), what correlation is (cosine of the angle between them), what the regression slope represents (scale $\mathbf{x}$ so it's as close as possible to $\mathbf{y}$), what the in-sample errors (or residuals) are (just the vector that when added to $\mathbf{x}b_1$ gives back $\mathbf{y}$), how multiple regression works, and much more. It is especially useful for understanding how R^2 relates to the correlation coefficient in the simple case and how its meaning changes in the multiple regression case, and why we use adjusted R^2. For the full presentation, see my 360 lectures here, but I prefer to just do this on the board.

For this set of notes, I will use only lowercase Roman letters for sample estimates to keep the notational burden light. I also hope that this encourages you to use simpler, less-fancy notation if it helps you remember what is what (admittedly, the Roman letter notation is potentially confusing).

0. I realized that last time I did not go over how to print regression lines on graphs, and many people struggled with doing so, or doing so efficiently. Fortunately, my final project do-file for SOC360 goes over exactly that, and this is yet another case where using locals or macros really simplifies things *once* you get over a bit of a learning curve. I'm copying and pasting the relevant section; see here for links to the whole thing. I left in my description of local macros designed for undergrads in case it is a useful reminder. So, let's start with this as a warm-up/transition from last week. You'll need to pull the GSS 2018 data first; see here. You can just run this and get a feel for how it works; then, go try to fix your code if you didn't do this with locals on the homework.

   ““

   - Graphs using the convenience of macros.
   - First, let's run the regression to get the results in Stata's memory.
   - We can add the prefix "qui" for quietly since we don't necessarily want
   - to see the output for this purpose.

qui reg educ paeduc

- Now we need to run the next part all in one ago – macros don't stick
- around in working memory beyond a single "run" of a .do-file or portion
- thereof. So, the contents of the local will be deleted after we run this.
- Macros are containers for numbers or text. The general syntax for making
- them for numbers is "local [macroname] = expression". Here, that lets us
- just store the return values from our analysis without having to write
- down actual numbers, which can be annoying.

local n = e(N) // the difference between r(N) and e(N) is a computer sciencey thing. matrix coeffs = e(b) // the coefficients are a vector so we store them like so. // Now we'll pull out those elements and put them in macros while // also rounding for readability. local slope = round(coeffs[1, 1], .01) * This puts the first coefficient, which has matrix coordinates [1, 1] * in a local called "slope" while also rounding it to the hundredths' * place. We'll do the same for the intercept. local intercept = round(coeffs[1, 2], .01) local r2 = round(e(r2), 0.01) * R^2 is just a scalar still.

- We print the contents of macros like this:

di `n' dislope'

```
* we use a left tick (next to the "!/1" key) and a right-tick to enclose
* the macro. Otherwise, nothing will happen (at best; at worst, an error)
```

- Finally, we'll put it all together. Note that we'll need line extenders,
- which are the three forward slashes. I'll also turn the legend off since
- it is typically not that useful. Note a couple of other small things, e.g.
- that to get italics or other special fonts, we write things like {it: text}.

scatter educ paeduc, jitter(5) mcolor(red) xtitle("Father's education (years)") /// ytitle("Respondent's education (years)") legend(off) /// title("Relationship of father's and child's education", size(medium)) /// note("Source: GSS 2018. {it:n} = `n'") ///  text(5 15 "Estimated regression equation" ///  "{it:Y} =slope'{it:X} + `intercept'" /// "{it:R} {superscript:2} =r2'", box fcolor(white)) || lfit educ paeduc

- Here's a sample advanced technique with density curves.
  - This example is inspired by Germán Rodriguez: https://archive.ph/pC17P

kdensity educ, gen(edX edY) nograph * This suppresses the graph but stores the X and Y coords in a new * variable that we'll use in a moment. kdensity paeduc, gen(paedX paedY) nograph

gen zero = 0 // we'll use this in a moment, don't worry.

twoway (rarea edY zero edX, color(red%30) ytitle("Density") /// title("Distribution of education in the US", size(medium)) xtitle("Years"))

```
* So, what we did here was make Stata shade the region of a graph between
* "edY" (the height of the density curve) and "zero" (just the X-axis)
```

```
* over the values of a third variable edX. For the sake of not having
* the graph too dark, we print the color red at only 30 percent. The
* rest is review. Finally, we can add father's education and some of
* the other key information.
```

qui sum paeduc, d local dadn = r(N) qui sum educ, d local rn = r(N)

twoway (rarea edY zero edX, color(red%30) ytitle("Density") /// title("Distribution of education in the US", size(medium)) xtitle("Years")) /// (rarea paedY zero paedX, color(green%30) /// note("Source: GSS 2018." /// "{it:n}{subscript:father} = dadn'; {it:n}{subscript: respondent} =rn'") /// legend(pos(9) ring(0) region(lcolor(black)) order(1 "Respondent" 2 "Father")))

```
* Note that "legend(order(1 "text1" 2 "text2" ...))" allows direct
* manipulation of the items in the legend. "Pos(clockface)" puts the
* legend at a certain "clocktime" orientation; "ring(0)" puts the
* legend in the plot; "region(lcolor(black))" puts a box around it.
```

````
````

1. This week, we're talking mostly about multiple regression. Stata syntax for multiple regression is extremely simple. Just add more predictors.

```
* reg y x1 x2
* e.g.
sysuse auto, clear
reg weight length turn
```

   Practice interpreting the slopes with a partner. There are two ways. First, you can take the conditional means approach that Gordon takes: fixing x_2 at some particular value, b_1 is the effect of a unit shift in x_1. Second, you can think about it more abstractly in the language of controlling: adjusting for the effect of x_2, this is the effect of x_1.

2. Gordon omits the formula for the multiple regression slopes but does show the formula for the standard error, which I think is a bit odd. Both are useful. The homework asks you to consider how the standard error changes, which is actually parallel to how the slope changes.

   Recall that for simple regression $b_1 = \frac{c_{x_1,y}}{s^2_x}$. Now, note that for regression with two variables…

$$b_1 = \frac{c_{x_1,y}s^2_{x_2} - c_{x_1,x_2}c_{y,x_2}}{s^2_{x_2}s^2_{x_1} - c^2_{x_1,x_2}}$$

   The exact details of the formula—derived in the appendix—are less important to memorize than to examine. How do multiple regression slopes work? Is multiple regression just a bunch of simple regressions? In what sense does the result of multiple regression for, say, b_1 "control" for the effect of $x_2, x_3, ...x_k$?

You can check that this formula is correct.

```
sysuse auto, clear
corr weight length turn , cov
matrix list r(C)
local c_x1y = r(C)[2,1]
di `c_x1y'
local v_x1 = r(C)[2,2]
local v_x2 = r(C)[3,3]
local c_x2y = r(C)[3,1]
local c_x1x2 = r(C)[3,2]
di ((`c_x1y'*`v_x2') - (`c_x1x2'*`c_x2y')) ///
 / ((`v_x1'*`v_x2') - (`c_x1x2')^2)
```

The formula for the whole vector of slopes, $\mathbf{b}$ (boldface means a vector, and a vector is just a list), is given in matrix algebra by the following. Don't panic; I'll explain. The first line says that we want our residuals (in-sample error) vector to be orthogonal (perpendicular) to the space of the predictors; this means that their matrix product is zero (by the law of cosines). Why would we want that? Two reasons: first, that means that the residual vector is unrelated to the xs, which is an assumption of our model (that no relevant variables have been left out that "lurk" in the residual"). The second is that we want to make our residual vector as small as possible; the way to do so is to make the projection of $\mathbf{y}$ down into the plane defined by $\mathbf{X}$ a straight line, which is the shortest path. You can see the appendix for details.

The rest is just algebra.

$$\begin{aligned}
\mathbf{X}^T\mathbf{e} &= \mathbf{0} \\
\mathbf{X}^T(\mathbf{y} - \mathbf{X}\mathbf{b}) &= \mathbf{0} \\
\mathbf{X}^T\mathbf{y} &= \mathbf{X}^T\mathbf{X}\mathbf{b} \\
\mathbf{b} &= (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}
\end{aligned}$$

What's that last line mean? It's instructive to write it for some particular b_k.

$$\begin{aligned}
b_k &= (\mathbf{X}_k^T\mathbf{X}_k)^{-1}\mathbf{X}_k^T\mathbf{y}_k \\
&= \sum_{j=1}^{n} c^{-1}_{x_j,x_k} c_{x_j,y}
\end{aligned}$$

Basically, any slope b_k is just the sum of the covariances of all the xs with y, inverted in each case by the covariance of some $x_j : j \neq k$ with x_k (and where $x_j = x_k$, it is just scaled by the variance of x_k).

3. Below is a small data-set. Practice finding the correlation purely by hand (I might even do this literally with pen and paper) using the techniques we saw last week, except with a twist. Find R^2 two ways; first, find it by finding the literal correlation coefficient. Second, find it by finding the model sum of squares and the total sum of squares. With only three observations, this is pretty easy to do by hand. **Hint**: if you center the variables, this makes things much easier. For example, then $r = \frac{\overline{xy}}{\sqrt{\overline{x^2} \cdot \overline{y^2}}}$, and most other quantities become very simple. *Also*, there is no intercept in the model (why?) and thus $\hat{y}_j = b_1 x_j$, which is very simple to find. (All of this is yet another reason to use centered models!)

   Time-permitting, when you are done, try to sketch the result using *centered* vectors in subject space. I'll post an answer after class since this one isn't obvious.

	x	y
0	9	15
1	15	0
2	6	8

4. Return to our model from above with `weight` regressed on `length`, and `turn`. Do this in Stata. Then, verify that R^2 is *both* $\frac{SS_M}{SS_T}$ *and* the square of $r_{\hat{y},y}$. *Discuss how R^2 incorporates the same information as the conditional and unconditional standard deviation but relates them more exactly.*

5. Discuss with a partner the reason for the use of the F-test. What are the problems associated with simultaneously testing many hypotheses about individual slopes?

6. For the above data, carry out the model F-test in steps (next time, I'll give some intuition for what the F-test represents geometrically). **Note that we do not need to include an intercept here since we have centered variables** (but generally, be careful when omitting an intercept). Below are our two models.

$$m_0 : Y = \beta_0 + \epsilon$$
$$m_1 : Y = \beta_0 + \beta_1 X_1 + \epsilon$$

   Can we get m_0 from m_1 by imposing a restriction on the coefficients of the latter?

   Now, our formula is…

$$\frac{SS_e^r - SS_e^f}{df^r - df^f} \cdot \frac{df^f}{SS_e^f}$$

   Note that it is helpful to think about the numerator df as being equal to the number of restrictions imposed on the null; that is what degrees of freedom really *are* (constraints on the space in which some quantity might lie). *How many restrictions do we need to impose on our model here?*

Helpfully, the sum of squared residuals with the restriction that $\beta_j = 0, j = 1, 2, ...p$ is always just the total sum of squares, which you already found, and the sum of squared residuals for the full model we also found already.

```
import numpy as np
import pandas as pd
np.set_printoptions(precision=3)
rng = np.random.default_rng(4301963)
y = rng.integers(low=0, high=20, size=3)
x = rng.integers(low=0, high=20, size=3)
y_c = y - np.mean(y)
x_c = x - np.mean(x)
xy = x_c*y_c
y_sq = y_c**2
x_sq = x_c**2
c_xy = np.mean(xy)
v_x = np.mean(x_sq)
v_y = np.mean(y_sq)
r_xy = c_xy / np.sqrt(v_x*v_y)
print(f"Hand calc of r is {r_xy}")
print(np.corrcoef([x,y]))
df = pd.DataFrame([x,y]).transpose()
df.columns = ["x", "y"]
print(df.to_markdown())
b_1 = c_xy/v_x
print(b_1)
yhat = b_1 * x_c
yhat_sq = yhat**2
v_yhat = np.mean(yhat_sq)
print(v_yhat/v_y)
print(r_xy**2)
e = y_c - yhat
e_sq = e**2
y_c_sq = y_c**2
F = (np.sum(y_c_sq) - np.sum(e_sq))*(2/np.sum(e_sq))
F
```

```
Hand calc of r is -0.6832434704397569
[[ 1.    -0.683]
 [-0.683  1.   ]]
|    |   x |   y |
|---:|----:|----:|
|  0 |   9 |  15 |
|  1 |  15 |   0 |
|  2 |   6 |   8 |
-1.119047619047619
0.46682163989856307
0.4668216398985629
```

```
1.751089972255252
```

2 Appendix

2.1 The formula for the regression slopes b_1, b_2 in regression on two continuous variables

Recall that we found the formula for the simple regression slope by assuming that we had vectors in n-dimensional space representing centered versions of our variables (picture in my previous notes; I will also draw on the board). Then, to make the prediction vector with typical component $b_1(x_j - \bar{x})$ perpendicular or orthogonal to the in-sample errors, which also makes them as short or as small as possible (so, this is a way of minimizing the sum of squares without directly using calculus!).

If we have two predictors, we have …

$$
\begin{aligned}
\mathbf{x}_1 \cdot \mathbf{e} &= \mathbf{0} \\
\mathbf{x}_2 \cdot \mathbf{e} &= \mathbf{0} \\
\mathbf{x}_1 \cdot (\mathbf{y} - [b_1\mathbf{x}_1 + b_2\mathbf{x}_2]) &= \mathbf{0} \\
\mathbf{x}_2 \cdot (\mathbf{y} - [b_1\mathbf{x}_1 + b_2\mathbf{x}_2]) &= \mathbf{0} \\
(b_1\mathbf{x}_1 \cdot \mathbf{x}_1 + b_2\mathbf{x}_1 \cdot \mathbf{x}_2) &= \mathbf{x}_1 \cdot \mathbf{y} \\
(b_1\mathbf{x}_1 \cdot \mathbf{x}_2 + b_2\mathbf{x}_2 \cdot \mathbf{x}_2) &= \mathbf{x}_2 \cdot \mathbf{y} \\
b_1 &= \frac{\mathbf{x}_1 \cdot \mathbf{y} - b_2\mathbf{x}_2 \cdot \mathbf{x}_1}{\mathbf{x}_1 \cdot \mathbf{x}_1} \\
b_2 &= \frac{\mathbf{x}_2 \cdot \mathbf{y} - b_1\mathbf{x}_2 \cdot \mathbf{x}_1}{\mathbf{x}_2 \cdot \mathbf{x}_2} \\
b_1 \left\{ 1 - \frac{(\mathbf{x}_2 \cdot \mathbf{x}_1)^2}{(\mathbf{x}_2 \cdot \mathbf{x}_2)(\mathbf{x}_1 \cdot \mathbf{x}_1)} \right\} &= \frac{(\mathbf{x}_1 \cdot \mathbf{y})(\mathbf{x}_2 \cdot \mathbf{x}_2) - (\mathbf{x}_2 \cdot \mathbf{x}_1)(\mathbf{x}_2 \cdot \mathbf{y})}{(\mathbf{x}_2 \cdot \mathbf{x}_2)(\mathbf{x}_1 \cdot \mathbf{x}_1)} \\
b_1 \left\{ \frac{(\mathbf{x}_2 \cdot \mathbf{x}_2)(\mathbf{x}_1 \cdot \mathbf{x}_1) - (\mathbf{x}_2 \cdot \mathbf{x}_1)^2}{(\mathbf{x}_2 \cdot \mathbf{x}_2)(\mathbf{x}_1 \cdot \mathbf{x}_1)} \right\} &= \frac{(\mathbf{x}_1 \cdot \mathbf{y})(\mathbf{x}_2 \cdot \mathbf{x}_2) - (\mathbf{x}_2 \cdot \mathbf{x}_1)(\mathbf{x}_2 \cdot \mathbf{y})}{(\mathbf{x}_2 \cdot \mathbf{x}_2)(\mathbf{x}_1 \cdot \mathbf{x}_1)} \\
b_1 &= \frac{(\mathbf{x}_1 \cdot \mathbf{y})(\mathbf{x}_2 \cdot \mathbf{x}_2) - (\mathbf{x}_2 \cdot \mathbf{x}_1)(\mathbf{x}_2 \cdot \mathbf{y})}{(\mathbf{x}_2 \cdot \mathbf{x}_2)(\mathbf{x}_1 \cdot \mathbf{x}_1) - (\mathbf{x}_2 \cdot \mathbf{x}_1)^2} \\
b_1 &= \frac{c_{x_1,y}s^2_{x_2} - c_{x_1,x_2}c_{y,x_2}}{s^2_{x_2}s^2_{x_1} - c^2_{x_1,x_2}}
\end{aligned}
$$

We can generalize this easily with matrix algebra.

$$
\begin{aligned}
\mathbf{X}^T\mathbf{e} &= \mathbf{0} \\
\mathbf{X}^T(\mathbf{y} - \mathbf{X}\mathbf{b}) &= \mathbf{0} \\
\mathbf{X}^T\mathbf{y} &= \mathbf{X}^T\mathbf{X}\mathbf{b} \\
\mathbf{b} &= (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}
\end{aligned}
$$

In words, this means that we first find the variance-covariance matrix for our predictors. Note that that is what $\mathbf{X}^T\mathbf{X}$ is if we divide it by $n-1$; if we do this to $\mathbf{X}^T\mathbf{y}$, this will cancel out when we multiply by the inverse, just like in the single-variable case. Then, we find the inverse, which is essentially the generalized form of "dividing by" something (we can't actually divide by a matrix because regular division only really makes sense with scalar quantities). So, we have a square p x p matrix which has the "undoing" of the variance of the xs on the main diagonal and the "undoing" of all the pairwise covariances on the off-diagonals. To get any particular b_k, we just take the covariance of y with all of the xs, which yields a p x 1 vector. Then, we dot that vector with a vector from the inverse matrix. The covariance of our b_k of interest with y is multiplied by the inverse of the variance of x_k, and then we add all pairwise covariances of $x_i, i \neq j$ with y but multiplied by the inverse of $x_j, x_k, j \neq k$.